MORRISON & FOERSTER LLP
MICHAEL A. JACOBS (Bar No. 111664)
mjacobs@mofo.com
MARC DAVID PETERS (Bar No. 211725)
mdpeters@mofo.com
DANIEL P. MUINO (Bar No. 209624)
dmuino@mofo.com
755 Page Mill Road
Palo Alto, CA 94304-1018
Telephone: (650) 813-5600 / Facsimile: (650) 494-0792

BOIES, SCHILLER & FLEXNER LLP
DAVID BOIES (Admitted *Pro Hac Vice*)
dboies@bsfllp.com
333 Main Street
Armonk, NY 10504
Telephone: (914) 749-8200 / Facsimile: (914) 749-8300
STEVEN C. HOLTZMAN (Bar No. 144177)
sholtzman@bsfllp.com
1999 Harrison St., Suite 900
Oakland, CA 94612
Telephone: (510) 874-1000 / Facsimile: (510) 874-1460

ORACLE CORPORATION
DORIAN DALEY (Bar No. 129049)
dorian.daley@oracle.com
DEBORAH K. MILLER (Bar No. 95527)
deborah.miller@oracle.com
MATTHEW M. SARBORARIA (Bar No. 211600)
matthew.sarboraria@oracle.com
500 Oracle Parkway
Redwood City, CA 94065
Telephone: (650) 506-5200 / Facsimile: (650) 506-7114

Attorneys for Plaintiff
ORACLE AMERICA, INC.

UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA
SAN FRANCISCO DIVISION

ORACLE AMERICA, INC.

Plaintiff,

v.

GOOGLE INC.

Defendant.

Case No. CV 10-03561 WHA

ORACLE AMERICA, INC.'S OPPOSITION TO GOOGLE'S MOTION FOR SUMMARY JUDGMENT ON COUNT VIII OF ORACLE'S AMENDED COMPLAINT

Date: September 15, 2011
Time: 2:00 p.m.
Dept.: Courtroom 9, 19th Floor
Judge: Honorable William H. Alsup

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Page(s)

CASES

STATUTES

OTHER AUTHORITIES

INTRODUCTION

The copying in this case is undisputed. Over a period of many months, Google employees and contractors sat down and duplicated, line by line, the specifications for Oracle's application programming interfaces ("APIs") for Java. When they were finished, they had reproduced specifications for 37 APIs from Java's core libraries that were identical, or nearly identical to Oracle's, and they had copied those specifications into Android code.

APIs are one of the most important aspects of a complex modern software program like Java. The Java APIs, in particular, serve as the guide to, and set forth the structure of, an extensive set of class libraries that provide developers with pre-packaged code they call upon during their programming. The choice of what to include in the APIs and how to arrange them requires creativity and skill. The 37 APIs contain thousands of different elements, arranged in a unique structure, with many interdependent relationships. They readily meet the standard for copyright protection. Google, in fact, claims copyright protection for its own APIs.

Google nonetheless urges the Court to hold that all APIs are not copyrightable as a matter of law. No court has ever done so. Google's request is contrary to Ninth Circuit law, which provides that the copyrightability of the non-literal components of a computer program is to be examined on the particular facts of each case.

Google also asks the Court to excuse its copying on the ground that it was required for compatibility. But Google *undermined* the compatibility of Java. It took the APIs it wanted, to attract developers to Android and gain market share quickly, and left those it did not ─ fragmenting Java and its "write once, run anywhere" promise.

Google also engaged in line-for-line copying of Oracle's source code, object code and comments in 12 separate programs. Google says it should be given a pass, claiming the copying is de minimis. It is not. Each program is entitled to protection as a separate work, and Google copied a substantial portion of each – and in most cases the entire program. And when combined with the copying of the APIs, Google's copying is significant even for Java as a whole.

STATEMENT OF FACTS

Google's motion does not dispute copyright ownership or that copying took place. And the parties' experts agree that copying the APIs had great value to Google – Google's expert goes so far as to say that Google's copying was "essentially required." (ECF No. 262-1, Astrachan Ex. 1 ¶¶ 130-136.) But the parties dispute many other points, including: whether the APIs contain copyrightable expression; the substantial similarity of the API specifications; the significance of Google's copying of the APIs and code; whether Google was required to copy the APIs for compatibility purposes; whether Android should be considered a transformative use; the impact on the potential market for and value of, Oracle's copyrighted work; and the propriety of Google's conduct. These disputed issues preclude summary judgment.

The Importance of APIs and Class Libraries. The Java programming language has been consistently ranked among the leading programming languages since its release in 1996. According to one widely used index, Java has been the top programming language since at least 2002. (*See* Mitchell Decl. Ex. 1 ("Mitchell Opening Report") ¶¶ 56-62.) There are over 6.5 million Java programmers worldwide. (*See* Swoopes Decl. Ex. 20,)

The Java platform's success is in part due to the popularity of the Java programming language. But along with the programming language, Sun developers wrote extensive, well-crafted APIs for many different functions. Another important factor in Java's success has been the "robust and elegant API specifications and class libraries that implement them." (Mitchell Decl. Ex. 2 ("Mitchell Opp. Report") ¶ 20.) As Joshua Bloch, a former Sun engineer now working for Google, who designed many of the Java APIs at issue, put it: "I find it very rewarding to design great APIs and have people come to me years later and say, wow, you know, the collections framework changed my life." (Swoopes Decl. Ex. 1 at 92:13-23.)

The APIs in question support Java's "class libraries." The class libraries Sun wrote help developers program more quickly and efficiently by enabling them to call upon certain pre-written "classes," so that they do not have to write them from scratch. (Mitchell Opening Report ¶¶ 51-52.) The APIs tell the programmer how to use the library, and include a set of names that can be used to access different features of the library, together with conventions about their use.

Java's class libraries include thousands of "methods" that are accessible through the APIs. Java's APIs and class libraries were designed to be extensive from the start. In 1996, for example, the libraries were documented in a 1650 page book called The Java™ Class Libraries: An Annotated Reference. (*Id.* ¶ 51.) These libraries have been become even larger over time.

The APIs Contain Many Original and Creative Elements. The Java APIs are the blueprint to the class libraries. They are not just a list of names and methods, but an extraordinarily complex structure of hierarchy and interdepency.

The design of the Java APIs includes methods, and "method signatures," which include the parameters associated with the method. There are also, in many cases, "fields," which are structures for storing data. These methods and fields are organized into classes, and the classes are organized into packages. For each class in the library, the API describes the fields and methods within it and identifies those that are exposed to other classes. (*See, e.g., id.* ¶ 171; Mitchell Opp. Report ¶¶ 20-28.) The API also defines the relationship of each class to other classes and packages of classes. The API designer must decide which packages, classes, methods and fields to include in the API, as well as the relationships and interdependencies among the thousands of individual elements. (*See id.*)

Designing the APIs for a complex structure like the Java platform requires a great deal of creativity and skill. In the words of Joshua Bloch: "API design is an art, not a science." (Swoopes Decl. Ex. 19. *See also* Ex. 1 at 235:3-20 (discussing "aesthetic" nature of API design).) Robert Lee, the leader of Android's core library development team, concurs that designing APIs is "absolutely" a creative activity. (Ex. 3 at 13:9-14:11.) Many articles and books are dedicated to the techniques and considerations involved in writing APIs. (Mitchell Opp. Report ¶¶ 6-9.) If APIs are written well, they create a beneficial cycle. A well-designed API is easier to learn and use, so developers will be more likely to adopt the platform and write more applications, creating more appeal to the platform for end users, which in turn attracts more developers. (*Id.* ¶ 172.)

A recent article in "Communications of the ACM," a pre-eminent journal, describes the creative challenges in API design: "There seems to be something elusive about API design that, despite years of progress, we have yet to master." It continues:

> *Good APIs are hard.* We all recognize a good API when we get to use one. Good APIs are a joy to use. They work without friction and almost disappear from sight: the right call for a particular job is available at just the right time, can be found and memorized easily, is well documented, has an interface that is intuitive to use, and deals correctly with boundary conditions.

(Swoopes Decl. Ex. 21, M. Henning, "API Design Matters," Communications of the ACM, Vol. 52 No. 5 at 46-47 (emphasis in original); Mitchell Opp. Report ¶ 6.)

Google Makes The Strategic Decision To Center Android On Java. Google decided that Java was by far the best choice for Android. It knew that Android's success would depend on providing consumers with a wide variety of user-installable applications, many of which would be developed by third parties. Allowing developers to program in Java would immediately give Google access to millions of developers. (*See, e.g.*, Ex. 10, GOOGLE-01-00025376 at 419 ("Strategy: Leverage Java for its existing base of developers").)

Attracting a pre-existing developer community was all the more important to Google because it was late to market and time was of the essence. (*See* Ex. 9, GOOGLE-01-00019529 at 530.) Google believed using Java would greatly shorten its time to market and "[d]ramatically accelerates our schedule." (Ex. 12, GOOGLE-14-00042244 at 246; Ex. 4 at 172:7-173:14, 176:5-177:11.) Copying the core Java APIs saved Google "an enormous amount of time." (Mitchell Opp. Report ¶ 80.) And Google's expert opines that it would have been "very difficult for Google" to persuade developers to switch to different APIs. (Astrachan Ex. 1 ¶ 135.)

Google Chooses To Base Its Android Platform On Java Without Taking A License. Oracle offers to license developers to make their own independent implementations of the Java API specifications provided they are fully compatible and do not fragment the platform. (*See, e.g.,* Swoopes Decl. Ex. 22; Ex. 27 at 21:14-22:13, 70:19-71:17.)

Google could have taken the standard license, or it could have negotiated to obtain a special license. It did neither. Google recognized internally it had two alternatives to working with Sun: abandon the work on "making Java central to our solution" and instead use the Microsoft platform and the C# language, or "*Do Java anyway and defend our decision, perhaps making enemies along the way.*" (Ex. 8, GOOGLE-01-00019527 at 527-528 (emphasis added).)

Google decided to roll the dice and push ahead with Java.

Google Copied The Java API Specifications. Google got down to work and began implementing not only the Java programming language and overall "write once, run anywhere" architecture, but also copying the core Java API specifications, including the 37 asserted here.[1]

Google admits that Android developers had access to the API specifications for Java 2 Standard Edition Version 5.0. (Ex. 7, Google Resp. to Pl.'s Req. for Admission No. 168.) Robert Lee, the lead developer for the Android core libraries, testified that he "consulted Sun's website for the API specifications when doing the work for Google" and that he assumed Google's contractor, Noser Engineering, did as well. (Ex. 3 at 65:8-66:16.) Mr. Lee saw that there were copyright notices on the specifications, but proceeded anyway, without consulting an attorney. (*Id.* at 66:17-67:5.; *see also* Ex. 2 at 161:5-20.)

Google does not deny that it deliberately copied 37 APIs from Java's core libraries. The APIs are extensive. They include thousands of elements. When fully printed out, they extend to more than 10,000 pages. (Swoopes Decl. ¶ 28.) When Google finished its copying, it proclaimed on the Android developer website: "Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language." (Ex. 26.)

Java and Android organize the 37 packages and classes identically. These 37 packages have identical names and identical or nearly identical structures. (Mitchell Opening Report ¶ 203.) For example, in both Java and in Android, the package java.nio contains the same ten classes, with identical names. (*Id.* ¶ 204.) Drilling down a level further into just one of these ten classes, java.nio.IntBuffer, shows that of the 25 individual methods that are included, 24 are also present in Android. Fifteen of the methods are identical, five make minor changes to the names but are otherwise identical, and four merely add the keyword "final" but are otherwise identical. (*Id.* ¶¶ 205-206; *see also* Exs. Copyright-A-D (side-by-side comparisons of 4 packages).) Google does not deny that it intentionally chose its names and organizational structure to match the Java APIs. (*See* Google Mot. at 16:15-22; Astrachan Ex. 1 ¶ 130.)

[1] Oracle is not asserting infringement of 14 other packages that appear in both Java and Android that were copied by Google because Oracle uses these packages under license from third parties, or allows third parties to use these packages under permissive terms.

The Java API specifications also include explanatory comments describing each method. Dr. Mitchell found that these comments are also substantially similar in two files in the Java and Android documentation. (Mitchell Opening Report ¶ 207.) A side-by-side comparison is included as exhibits E and F to Dr. Mitchell's opening report. (*See id.* at Exs. Copyright-E-F.)

Android's Source Code Is Derived From The Copied API Specifications. Google also implemented the copied API specifications into Android source code. Google tries to downplay the significance of this, acknowledging only that it copied the "names of packages and methods and definitions." (Google Mot. at 16:15-17.) In fact, Google copied the entire hierarchical and organizational structure of the APIs into the Android source code.

Mr. Lee, Android's lead core library developer, wrote in a 2008 document that he was tasked with "re-implementing" the Java APIs. (Swoopes Decl. Ex. 14, GOOGLE-40-00034698.) Mr. Lee testified that Google worked to "implement core libraries according to the Java APIs." (Ex. 3 at 14:7-11, 64:8-20, 65:8-66:16.) Google copied Java API specifications verbatim, or nearly verbatim, into its source code and then included code that implemented what was called for by the specification. (*See* Mitchell Opening Report ¶¶ 209-219.) Google repeated this thousands of times, until it had replicated all, or nearly all, of the methods contained in the 37 API specifications, in accordance with the same design. (Mitchell Opp. Report ¶ 77.) Instead of determining for itself what to include in the APIs and class libraries, and their associated interdependencies, Google simply copied the Java APIs and wrote code to carry out each of the specified functions, saving an enormous amount of time. (*Id.* ¶ 80.) The source code derived from the API specifications can also be found on Android devices, compiled into executable form by Google and device manufacturers. (*See* Mitchell Opening Report ¶¶ 220-223.)

Google Literally Copied Oracle's Source Code, Object Code and Comments. Two Android source code files contain lines of code for a method called "rangeCheck" that are identical to those in Oracle's copyrighted java.util.Arrays.java program. (Mitchell Opening Report ¶ 233.) The table below shows a side-by-side comparison:

rangeCheck() from Oracle java.util.Arrays.java lines 1318-326 [spacing adjusted for comparison]	rangeCheck() from Android TimSort.java lines 915-923 [spacing adjusted for comparison]

private static void rangeCheck(int arrayLen, int fromIndex, int toIndex) {
if (fromIndex > toIndex)
throw new IllegalArgumentException("fromIndex(" + fromIndex +
") > toIndex(" + toIndex+")");
if (fromIndex < 0)
throw new ArrayIndexOutOfBoundsException(fromIndex);
if (toIndex > arrayLen)
throw new ArrayIndexOutOfBoundsException(toIndex);
}
	private static void rangeCheck(int arrayLen, int fromIndex, int toIndex) {
	if (fromIndex > toIndex)
	throw new IllegalArgumentException("fromIndex(" + fromIndex +
	") > toIndex(" + toIndex+")");
	if (fromIndex < 0)
	throw new ArrayIndexOutOfBoundsException(fromIndex);
	if (toIndex > arrayLen)
	throw new ArrayIndexOutOfBoundsException(toIndex);
	}

Id. The author of Oracle's Arrays.java code is former Sun engineer Joshua Bloch, who also wrote the code for Android's rangeCheck. (*See* Swoopes Decl. Ex. 1 at 39:3-5, 162:8-163:7, 174:1-12.) When asked whether he accessed the Oracle source code while working on TimSort, Bloch testified that there "is a strong indication that it is likely that I did." (*Id.* at 181:7-14.)

Google copied Oracle's copyrighted Java object code in 8 additional files in their entirety. Using a decompiler, Google used the Java object code to generate source code for Android. (Mitchell Opp. Report ¶¶ 89-90; Mitchell Opening Report ¶ 242.) The source code for these eight Android programs is nearly identical on a line-by-line basis to files decompiled from Oracle files JDK5, which is available for download from Oracle's website. (Mitchell Opening Report ¶¶ 241-243, Exs. Copyright-J-Q (side by side comparison of 8 files).

Google does not dispute the copying. But the parties have a factual dispute over its significance. Google's expert, Dr. Astrachan, states that the 8 copied files are insignificant "test" files that are in part "dummy files." (Astrachan Ex. 1 ¶¶ 163-165.) Oracle's expert, Dr. Mitchell, compared these files to the Oracle code and concluded they are not simply dummy test files. The Oracle code is not in the "test" portion of Oracle's directories. To the extent Google does them for testing, test files are significant too. (Mitchell Opp. Report ¶¶ 91-95.)

The copying did not stop there. Two more Android files contain a series of comments that are nearly identical to comments in corresponding Java files. (Mitchell Opening Report ¶ 249.) The comments are compared side-by-side at Mitchell Opening Report Exhibits Copyright R-S.

Android Gained Rapid Market Acceptance At Oracle's Expense. Google's strategy of centering Android on Java has been phenomenally successful. Android's share of the U.S. Smartphone market has jumped from just below 4% in Q4 '09, a year after its release, to about 30% of the market by early 2011. (*See* Mitchell Opening Report ¶¶ 70, 72-73.)

Google's success has come at Oracle's expense. Oracle's Chief Corporate Architect, Edward Screven, testified that "Android has basically foreclosed" Oracle's strategy for succeeding in the Smartphone market and that Java is "pretty well locked out of the smartphone market because of Android." (Swoopes Decl. Ex. 6 at 71:13-72:4, 111:10-112:2.)

In addition, Oracle has been harmed because Google fragmented the Java APIs, severely undercutting Java's "write once, run anywhere" promise. Though Google copied 37 Java APIs from Java's core libraries, there were many other APIs it did not implement. Despite its claim that its copying was required for compatibility, the reality is Google took only the parts it wanted and created many other, incompatible APIs for Android. As a result, many programs written in Java for other platforms will not run on Android, and many programs written for Android will not run on Java platforms and devices. (*See* Mitchell Opp. Report ¶¶ 101-113.)

Notably, Google requires its OEMs to maintain the full set of Android APIs – *including the 37 APIs it copied from Oracle* ─ to prevent fragmentation of the Android platform. Android's license is similar to Java's providing, among other things, that "[d]evice implementations MUST NOT omit any managed APIs," "MUST NOT modify the publicly exposed APIs on the Android platform," and "MUST NOT add any publicly exposed elements…to the APIs." (Mitchell Opp. Report ¶ 115.) Google itself, of course, has done all of these things to the Java APIs.

ARGUMENT

I. GOOGLE'S WHOLESALE COPYING AND IMPLEMENTATION OF JAVA API SPECIFICATIONS IS COPYRIGHT INFRINGEMENT.

No court has ever found that the APIs for a complex software platform like Java are ineligible for copyright protection. Google asks this Court to find that all application programming interfaces are uncopyrightable as a matter of law ─ *regardless of their expressive content* ─ because they are supposedly "methods of operation" that are simply "functional

requirements for compatibility." (*See, e.g.*, Google Mot. at 1, 13-14.) Google's request is not just unprecedented, it is contrary to Ninth Circuit law, which calls for an examination of the particular facts of each case, to distinguish unprotectable ideas from copyrightable expression. The Java APIs that Google copied are complex, creative, and expressive. They are not simply dictated by functionality or compatibility requirements. They pass the test for copyrightability in this Circuit.

A. Ninth Circuit Law Requires An Analysis Of The Particular Expressive Content Of A Copyrighted Computer Program.

Computer software is expressly subject to copyright protection. 17 U.S.C. § 101. "[C]opyright protection extends not only to the 'literal' elements of computer software – the source and object code – but also to a program's nonliteral elements, including its structure, sequence, organization, user interface, screen displays and menu structures." *Merchant Transaction Sys., Inc. v. Nelcela, Inc.*, No. CV 02-1954, 2009 U.S. Dist. LEXIS 25663, at *29 (D. Ariz. Mar. 17, 2009) (quoting *Gen. Universal Sys., Inc. v. Lee*, 379 F.3d 131, 142 (5th Cir. 2004)). The Ninth Circuit does not automatically disqualify any of these elements of software from copyright protection. To the contrary, the Ninth Circuit has held:

> Whether the nonliteral components of a program, including the structure, sequence and organization and user interface, are protected depends on whether, on the particular facts of each case, the component in question qualifies as the expression of an idea, or an idea itself.

Johnson Controls, Inc. v. Phoenix Control Sys., Inc., 886 F.2d 1173, 1175 (9th Cir. 1989).

In analyzing the "particular facts," the Ninth Circuit stated in *Apple Computer, Inc. v. Microsoft Corp.* ("*Apple I*") that "[u]sing analytic dissection and, if necessary, expert testimony," the court should separate protectable expression from unprotectable ideas to determine whether a work is entitled to copyright protection. 35 F.3d 1435, 1443 (9th Cir. 1994).

This Court has recognized that, "[T]he scope of protection given to copyrighted software is still a developing area of law." *Miller v. Facebook*, No. C 10-00264 WHA, 2010 U.S. Dist. LEXIS 61715, at *13 (N.D. Cal. May 27, 2010). Google would have the Court skip analysis of whether the APIs contain copyrightable expression, and instead hold that APIs are not eligible for copyright protection because they are "unprotectable methods of operation." (*See* Google Mot. at 13-14.) Google's argument relies on the First Circuit's approach in *Lotus v. Borland*, suggesting

that, even if code is expressive, it is unprotectable if it can be characterized as a "method of operation." The Ninth Circuit has never adopted *Lotus'* approach, and other circuits have refused to follow it. Moreover, *Lotus* is factually distinguishable.

Lotus addressed the copyrightablity of the menu in Lotus 1-2-3. *See Lotus Dev. Corp. v. Borland Int'l, Inc.*, 49 F.3d 807 (1st Cir. 1995), *aff'd by an evenly divided court*, 516 U.S. 233 (1996).[2] "Accepting" the district court's finding that Lotus made expressive choices in designing the menu, the court concluded nonetheless that the menu was uncopyrightable because it was a "method of operation" under 17 U.S.C. § 102(b). *Id.* at 816. The court defined "method of operation" broadly, as "the means by which a person operates something." *Id.* at 815.

The *Lotus* court's definition of an unprotectable "method of operation" is perilously close to the definition of a computer program under the Copyright Act, which "is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result." 17 U.S.C. § 101. The exception proposed by *Lotus* threatens to swallow the rule whole.

In *Apple Computer, Inc. v. Formula Int'l, Inc.*, 725 F.2d 521, 523-24 (9th Cir. 1984), the Ninth Circuit rejected the argument that computer operating systems programs ─ programs "designed to manage the computer system" ─ are uncopyrightable because they are only "ideas" or "processes" under 17 U.S.C. § 102(b). Examining the Copyright Act and its legislative history, the court upheld the district court's finding that the Copyright Act "extends protection to all computer programs regardless of the function which those programs perform." *Id.* at 523.

The Ninth Circuit cited to the Final Report of the National Commission on New Technological Uses of Copyright Works (CONTU), which "was established by Congress to consider, *inter alia,* to what extent computer programs should be protected by copyright law," and whose recommended statutory changes were adopted "verbatim" by Congress:

> *The copyright status of the written rules for a game or a system for the operation of a machine is unaffected by the fact that those rules direct the actions of those who play the game or carry out the process.* Nor has copyright been denied to works simply because of their utilitarian aspects. . . .

[2] The affirmance of a decision by an equally divided Supreme Court is not precedential. *See Exxon Shipping Co. v. Baker*, 554 U.S. 471, 484 (2008).

> *That the words of a program are used ultimately in the implementation of a process should in no way affect their copyrightability.*

Id. at 524 (quoting CONTU Report at 21 (emphasis in Ninth Circuit opinion)). *See also Toro Co. v. R & R Prods. Co.*, 787 F.2d 1208, 1210-11 (8th Cir. 1986) (citing House Report); 1 *Nimmer on Copyright* § 2.03[D] (same).

Mitel, Inc. v. Iqtel, Inc., 124 F.3d 1366 (10th Cir. 1997), the only other case Google cites in support of its "method of operation" argument, explicitly rejects the *Lotus* approach. (Google Mot. at 13:26-14:8.) After discussing the *Lotus* decision at length, the Tenth Circuit stated:

> We conclude that although an element of a work may be characterized as a method of operation, that element may nevertheless contain expression that is eligible for copyright protection. Section 102(b) does not extinguish the protection accorded a particular expression of an idea merely because that expression is embodied in a method of operation at a higher level of abstraction.

Mitel, 124 F.3d at 1372. The court went on to analyze whether the plaintiff's 4-digit code hardware interface contained original, protectable expression and concluded it did not. *Id.* at 1373-76; *see also Brown Bag Software v. Symantec Corp.*, 960 F.2d 1465, 1477 (9th Cir. 1992) (suggesting "screens, menus and keystrokes" may be copyrightable); *Gen. Universal*, 379 F.3d at 142 (stating "menu structures" are copyrightable).

The *Lotus* case is also factually distinguishable. It concerned a consumer-friendly set of 50 menus with a simple hierarchy. 49 F.3d at 809. This type of simple user menu cannot compare to the Java APIs, which are "designed for programmers and are designed to provide and describe a very rich development environment." (Mitchell Opp. Report ¶ 18.) They contain thousands of elements, layers of complex interdepencies, data structures, and numerous expressive choices that are not dictated by function.

Moreover, even under the *Lotus* definition, the APIs are not "methods of operation" because a fundamental purpose of the APIs is not just to call a function, but to "impose a level of abstraction and structure on top of the underlying software development program" to help programmers understand its complexities. (*Id.* ¶ 23.) The computer does not care about this organizational structure. But humans need some form of order to handle complex programming tasks and work efficiently. (*Id.*) As Dr. Mitchell explains, "Some of today's software systems

are among the most complex artifacts ever created by man, and the use of APIs is the core structuring concept that software designers use to manage their complexity." (*Id.* ¶ 18.) They cannot be dismissed as "non-expressive" methods of operation.

B. Google Cannot Meet Its Burden Of Showing That The Java APIs Do Not Contain Copyrightable Expression.

The burden is on Google to show the APIs it copied do not contain copyrightable expression because Oracle registered the copyrighted works at issue, which include the API specifications and the code. (*See* Swoopes Decl. Ex. 18.) "In judicial proceedings, a certificate of copyright registration constitutes prima facie evidence of copyrightability and shifts the burden to the defendant to demonstrate why the copyright is not valid." *Biberro Sys., Inc. v. Colwell, Inc.*, 893 F.2d 1104, 1106 (9th Cir. 1990); s*ee also Swirsky*, 376 F.3d at 851 ("Because [the work] has a valid certificate of registration with the copyright office…[plaintiff] is entitled to a presumption of originality."). Google cannot meet this burden.

1. Copyright Law Protects the Original Expression in the Selection, Coordination and Arrangement Of the Java APIs.

Copyright law protects expression in software design, including the selection and structure of software elements. Google's motion glosses over this issue, but selecting what to include in an API, and designing the appropriate structure to contain it, takes a great deal of creativity and skill.

The purpose of the Java APIs is to provide developers with access to prewritten code in various areas to save them the time of having to write the code themselves. The APIs need not include any particular method, class or package (with the exception of a few classes closely tied to the Java language) for the Java language to function. (Mitchell Opp. Report ¶ 20.) The decision to include these elements is therefore not dictated by function, and allows for a great deal of creativity. If the designer includes too little, the developers will not have the tools and flexibility they prefer. If the designer includes too much, the APIs become overwhelming and difficult to use. (*Id.* ¶ 21.)

This creative process extends from the big decision of which packages and classes to include all the way to how many parameters to include with a particular method and what order to list them in. For example, the Java APIs provide six versions of the method java.lang.Runtime

exec() that differ only in their parameter lists. (Mitchell Opening Report ¶¶ 185-186.) While giving developers as many parameters as possible provides for greater flexibility, often the better choice is to establish a set order and limit the options. (Mitchell Opp. Report ¶ 22.) The API designer must make this type of creative choice at every level of the API.

In addition, the complex organizational structure of the API, while not important to the computer, is critical to the programmer. A key design choice is which classes of objects will inherit characteristics from others. Java classes are arranged hierarchically, so that a subclass may inherit characteristics from its parent. The API designer must choose this hierarchical structure and decide whether a class will be independent or should be defined from or exposed to another class. Classes can also call on other classes outside of their hierarchical structure. There are even more complicated types of interrelationships in the Java APIs. One example is an internal construct called Interface, which is essentially a form of class that relies on other classes to implement its objects. (*Id.* ¶ 19.) Interfaces are often used to "group classes that are similar in some respect but need not share implementation." (*Id.* ¶¶ 23-27.) Another important design feature of the APIs is "fields," which store data associated with the class, and which can themselves be made accessible to other methods or classes. (*Id.* ¶ 19.)

These design choices have important implications. As one author explains, "A well-designed API can be your organization's biggest asset. Conversely, a poor API can create a support nightmare and even turn your users toward your competitor." (Swoopes Decl. Ex. 17, M. Reddy, API Design for C++ 4 (2011). *See also* Mitchell Opp. Report ¶ 25.)

These design choices are handled differently by different API designers. Even for ideas such as storing and manipulating collections of data as a single unit ─ ideas common across many programming platforms ─ there are many expressive choices. (*See* Mitchell Opening Report ¶¶ 193-199.) The collections hierarchies for Java, C++, and Smalltalk, for example, all of which have the same purpose, follow very different organizational schemes and use different language to describe their elements, as is illustrated pictorially in Exhibit 4 to the Mitchell Declaration. (*Id.*; *see also* Mitchell Decl. Ex. 4 (comparative diagram).) The differences between the collections hierarchies represent the different expressive choices made by the API designers.

Google has cited no decision holding that the structure of a computer program with this level of complexity and interdepency is precluded from copyright protection. Decisions have, however, recognized the copyrightability of programs with much simpler structures. For example, in *Autoskill*, the Tenth Circuit upheld a district court's finding of likelihood of success on copyrightability of the "organization, structure and sequence" of a computer program designed to teach reading skills, including a "keying procedure" that required students to respond by pressing the 1, 2 or 3 keys. *Autoskill, Inc. v. Nat'l Educ. Support Sys., Inc.*, 994 F.2d 1476, 1492, 1495 n. 23 (10th Cir. 1993). Similarly, in *CMAX/Cleveland, Inc. v. UCR, Inc.,* 804 F. Supp. 337, 355 (M.D. Ga. 1992), the court held that the file structures for a software program design for companies in the "rent to own" business constituted copyrightable expression. *See also Eng'g Dynamics, Inc. v. Structural Software, Inc.*, 26 F.3d 1335, 1345-46 (5th Cir. 1994) (rejecting argument that input data formats in structural engineering software program did not contain original expression because they were purportedly "organized in a particular fashion to effectuate the performance of mathematical calculations").

Google understands the creativity and importance of APIs. It asserts copyright and other intellectual property rights over its own APIs. (Swoopes Decl. Exs. 23-25.) These rights were applied for and maintained during the tenure of long time Google CEO and current Chairman Eric Schmidt. Google quotes extensively from Mr. Schmidt's 1994 testimony advocating against copyrightability, identifying him only as the "Sun CTO." (*See, e.g.*, Google Mot. at 1.) As this shows, there is a distinction between what a person advocates the law should be and what it is.

2. Copyright Law Protects the Names of the Packages, Classes, Interfaces, Fields, and Other Elements in the Java APIs

Oracle is also entitled to copyright protection in the names of the many different elements that make up the Java APIs. Google argues there is no copyright protection in the *thousands of names* it copied by pointing to isolated examples. (Google Mot. at 17:16-19:2.) This cannot defeat the presumption that the names in the Java API specifications are original. *See Swirsky*, 376 F.3d at 851. Google's copying of the names of 37 packages, 458 classes, 158 Interfaces,

2,427 methods, 893 fields, and other elements (*see* Purdy Decl. Ex. A, Purdy Opening Report ¶¶ 13-17) did not come about due to the existence of limited language.

Dr. Mitchell has provided numerous counterexamples of names that are not dictated by compatibility or programming practices. (*See, e.g.,* Mitchell Opp. Report ¶¶ 40-41 (describing java.beans.Customizer, java.beans.Introspector and other examples).) Java's API designers often chose names according to style rules they themselves created, but frequently departed from those rules, choosing names for aesthetic reasons. (*Id.*; *see also* Mitchell Opening Report ¶¶ 179-187.)

Google cites *Merchant* for the proposition that "isolated field names" are not entitled to individual protection. (Google Mot. at 17:22-26 (quoting *Merchant*, 2009 U.S. Dist. LEXIS 25663, at *44).) But in that case the court *denied* summary judgment, holding that "the coordination, selection, and arrangement of these field names, as well as the fields and field types themselves . . . may be entitled to protection as a compilation so long as they 'are numerous enough and their selection and arrangement original enough that their combination constitutes an original work of authorship." *Merchant*, 2009 U.S. Dist. LEXIS 25663, at *46 (citation omitted).

Similarly, in *CMAX/Cleveland*, the plaintiff accused the defendant of copying file layouts and file names, among other elements. 804 F. Supp. at 337. After trial, the court ruled for the plaintiff, rejecting the argument that "the programs are only similar in their file names and the sequence of the field names within each file," holding that "selection and arrangement of the field definitions within the files . . . are the expression of an idea." *Id.* at 354. The selection and arrangement of the fields in the Java APIs are likewise expressive.

Applying the kind of microscopic focus Google does here would "result in almost nothing being copyrightable because original works broken down into their composite parts would usually be little more than basic unprotectable elements like letters, colors and symbols." *Boisson v. Banian, Ltd.*, 273 F.3d 262, 272 (2d Cir. 2001). Google's expert engages in just such an analysis, trying to calculate the average word length of *7,796* method names to claim they are not original. (*See* Astrachan Ex. 1 ¶ 114.) Claiming there is no originality in the selection of almost 8000 names borders on absurd. That is particularly true since, in the Ninth Circuit, "[a]ll that is needed

to satisfy originality is for the author to contribute 'something more than a merely trivial variation.'" *Swirsky*, 376 F.3d 841 at 851 (internal citation and quotation omitted).

Moreover, even if this Court were to find that some of Google's cherry picked examples of names are unoriginal, "a claim of copyright infringement can be based on infringement of a combination of unprotected elements." *Dream Games of Ariz., Inc. v. PC Onsite*, 561 F.3d 983, 988 (9th Cir. 2009). At a minimum, the issue of whether the thousands of names that Google copied were original is a triable issue of fact. *See id.* (triable issue as to originality of song).

C. The *Scenes A Faire* And Merger Doctrines Do Not Apply Here.

Google misapplies the *scenes a faire* and merger doctrines. The merger doctrine provides that when there are so few ways of expressing an idea that "an idea and its expression are indistinguishable, or 'merged,' the expression will only be protected against nearly identical copying." *Apple I*, 35 F.3d at 1444 (citations omitted). The "closely related" *scenes a faire* doctrine provides that when certain features are "as a practical matter indispensable, or at least standard, in the treatment of a given [idea], they are treated like ideas and are therefore not protected by copyright." *Id.* (internal quotation and citations omitted).

Neither doctrine applies here. The API designer has great liberty to choose what to include in the APIs; it is not necessary to include any particular method, field, class or package. (Mitchell Opp. Report ¶ 20.) The API designer also is at complete liberty to choose how to structure the APIs, because that is designed for the benefit of the programmer. (*Id.* ¶ 23.) Google cannot claim all the features of the APIs are "as a practical matter indispensable" or that the idea behind the APIs and the myriad ways of expressing them are "merged."

Google nonetheless argues that courts "have routinely emphasized that compatibility is a legitimate aim, and can override claims for infringement under either the *scenes a faire* or fair use doctrines." (Google Mot. at 15.) But Google's *scenes a faire* argument is backwards. The proper test under *scenes a faire* or merger is whether *Oracle* was constrained in its choice of design because of the need to be compatible with some other product or standard, not Google. *See Control Data Sys., Inc. v. Infoware, Inc.*, 903 F. Supp. 1316, 1323 (D. Minn. 1995) (question is whether external factors limited choices available to plaintiff's programmers, not defendant's).

Google argues the APIs are not expressive because it decided to copy certain features from them in order to be compatible. (Google Mot. at 16). But Google's choice to copy does not alter the APIs' expressive nature. Google cites the lower court's decision in *Mitel*, holding that the defendant's copying was justified by "efficiency reasons" and "dictated by external factors" because its technicians were used to the plaintiff's code. (*Id.*) But in upholding the lower court's decision for other reasons, the Tenth Circuit chastised the court for incorrectly applying the law, stating that the proper focus "should have remained upon the external factors that dictated [plaintiff's] selection of registers, descriptions, and values," rather than on whether external factors "justified [defendant's] copying." *Mitel*, 124 F.3d at 1375.

Baystate Techs. v. Bentley Sys., 946 F. Supp. 1079 (D. Mass. 1996) does take Google's approach, but cannot be squared with Ninth Circuit law on the scenes a faire and merger doctrine. *See Apple I*, 35 F.3d at 1444. Moreover that court is in the First Circuit, and was bound by the *Lotus* decision. Just as importantly, *Baystate,* and the other case Google cites, *Bateman v. Mnemonics*, 79 F.3d 1532 (11th Cir. 1996), are factually distinguishable. *Baystate* involved a much simpler data translator that converted between different file formats, thus requiring an identical data organization to plaintiff's program. *See* 946 F. Supp. at 1088. *Bateman* involved reverse engineering to create a new platform that was compatible with preexisting hardware and applications. *See* 79 F.3d at 1540. Google did not need to copy the APIs for compatibility with Java, nor did it do so. Rather, Google took the APIs it wanted and wrote its own in place of the rest, destroying compatibility through fragmentation. *Bateman* held that any finding relating to compatibility "will depend on the particular facts of a case," and remanded the district court's decision so that a jury could consider whether compatibility constrained expression. 79 F.3d at 1547 n.32 (11th Cir. 1996). Notably, the court held: "It is an incorrect statement of the law that interface specifications are not copyrightable as a matter of law." *Id.* at 1547.

D. Google Copied The APIs.

Google does not dispute that it copied the 37 Java API specifications into Android's API specifications. This violated Oracle's reproduction right. *See* 17 U.S.C. § 106(1). It then created a derivative work by deliberately copying the API specifications into the Android source code,

following their complex design to the letter. (*See* Swoopes Decl. Ex. 3 at 14:7-11, 65:8-66:16, 64:8-20; Mitchell Opening Report ¶¶ 209-29; Mitchell Opp. Report ¶¶ 77-81.) 17 U.S.C. §§ 101, 106(2).

Google claims that "some" of the Android core libraries "were in part derived" from Apache Harmony, an open source project. (Google Mot. at 5:14-6:8.) Even if true, they were also derived from Oracle. Android's lead core library developer has admitted that he "consulted Sun's website for the API specifications when doing the work for Google." (*See* Swoopes Decl. Ex. 3 at 65:8-66:16*; see also* Ex. 2 at 161:12-20.) In addition, Google was well aware of Sun and Apache's highly publicized dispute over whether Harmony code would be licensed for use in mobile devices, and so could not have believed using the Harmony code gave it any kind of mobile device license from Sun. (*See, e.g.*, Ex. 1 at 119:13-122:4.) But it does not matter in any event. A defendant is liable when he copies, with or without a license, from a third party who copied from the plaintiff. *See, e.g., Pye v. Mitchell*, 574 F.2d 476, 481 (9th Cir. 1978).

Google argues the standard of comparison should be "virtual identity" because the range of protectable expression is "narrow." (Google Mot. at 12:3-5 (quoting *Apple I*, 35 F.3d at 1439).) But the Java APIs have many highly creative and expressive elements and are not dictated by function. The appropriate standard is accordingly "substantial similarity."

Oracle prevails under either standard. Google copied the 37 Java API specifications verbatim or nearly verbatim, incorporating their same hierarchical structure. (*See* Mitchell Opening Report ¶¶ 200-208, Exs. Copyright-A-F.) Google apparently disputes the substantial similarity between the specifications. (*See* Google Mot. at 24.) If Oracle presents "'indicia of a sufficient disagreement concerning the substantial similarity of [the] two works,' then the case must be submitted to a trier of fact." *Swirsky*, 376 F.3d at 844. The expert testimony and documentation submitted by Oracle in support of this opposition more than satisfy that standard.

Google also gets the law wrong, comparing the copyrighted works with Android as a whole. (*See* Google Mot. at 24-25.) It does not matter what else Google added. "No plagiarist can excuse the wrong by showing how much of his work he did not pirate." *Shaw v. Lindheim*, 919 F.2d 1353, 1362 (9th Cir 1990) (quoting 4 *Nimmer on Copyright* § 13.03[B][1][a])).

E. Google's Copying Was Not Fair Use.

Google's copying cannot be excused as fair use. At best, Google's defense raises material questions of fact that cannot be decided on summary judgment.

Google's fair use argument relies heavily on the Ninth Circuit's decisions in *Sony* and *Sega* — two cases with very different facts that raise very different policy concerns. The focus of the court's inquiry in both cases was whether it was fair use to reverse engineer a copyrighted product where "disassembly is the only way to gain access to the ideas and functional elements embodied in a copyrighted [work]." *See Sega Enters., Ltd. v. Accolade, Inc.*, 977 F.2d 1510, 1527-28 (9th Cir. 1993); *Sony Computer Entm't, Inc v. Connectix*, 203 F.3d 596, 603-04 (9th Cir. 2000). But here, Oracle's APIs were in plain view for anyone to see, so there was no need to copy them to discover their functional elements.

Another key to both decisions was that they concern intermediate copying only. The final product was not alleged to infringe the copyright. *See Sega,* 977 F.2d at 1527-28; *Sony*, 203 F.3d at 604 n.7. As the Ninth Circuit noted in *Sega*, "[o]ur conclusion does not, of course, insulate [defendant] from a claim of copyright infringement with respect to its finished products." 977 F.2d at 1528. Here, of course, Oracles accuses the final Android APIs and code of infringement. The reasoning in *Sega* and *Sony* does not apply. The four statutory factors are discussed below.

Purpose and Character of Use – An express part of this first factor, which Google fails to address, is "whether [the] use is of a commercial nature or is for nonprofit educational purposes." 17 U.S.C. § 107(1). (*See* Google Mot. at 19-20.) Google's use is clearly commercial. This "tends to weigh against a finding of fair use." *Campbell v. Acuff-Rose,* 510 U.S. 569, 585 (1994) (citation omitted).

Citing *Sony*, Google argues that this factor nonetheless weighs in its favor because Android was a new smartphone platform so its work is "transformative." It is not. The relevant inquiry is whether the use "merely supersedes the objects of the original creation, or instead adds something new, with a further purpose or differing character, altering the first with new expression, meaning or message." *Campbell*, 510 U.S. at 579. In *Sony*, even though the defendant's product allowed users to play games designed for the Sony Playstation on a different

medium – the personal computer – the court found the work to be only "modestly transformative." 203 F.3d at 606. Google copied the Java APIs to use them the same way Oracle does by licensing an operating system to computer and mobile device manufacturers. The use has no transformative value. But if it has any at all, it is even less than the "modest" value in *Sony*.

When the allegedly transformative nature of Android is weighed against commercial use, this case comes out on the opposite side from *Sony*. In *Sony*, the Ninth Circuit concluded that the modestly transformative use was not outweighed by the fact that the use was commercial because: (1) there was no copyright infringement in the final product, so commercial use was only "indirect or derivative" and (2) it produced a product that "would be compatible" with Sony Playstation games. *Id.* at 607 (quoting *Sega*, 977 F.2d at 1522). Neither factor is present here.

First, the final product infringes here, so commercial use is not "indirect or derivative." *See Atari Games Corp. v. Nintendo of America, Inc.,* 975 F.2d 832, 843 (Fed. Cir. 1992) (fair use copying "must not exceed what is necessary to understand the protected elements" and "does not justify extensive efforts to profit from replicating expression") (applying Ninth Circuit law).

Second, unlike in *Sega* and *Sony,* Google did not need to copy to achieve compatibility, and did not produce a compatible product. In *Sega* and *Sony* the only way for defendants' programs to run was to copy the manufacturer's functional interface. *See, e.g., Sega*, 977 F.2d at 1515 (need to input four letter initialization code). Here, Google could have used the Java language without copying Oracle's APIs. With the exception of a very few classes, the Java APIs are not required to use Java at all. (Mitchell Opp. Report ¶¶ 57-60.) There are many examples of individuals or organizations writing their own APIs for other languages, like C++, as well as for Java. Google could have written its own APIs, and the proof of that is that Google did. Google wrote new APIs for the Android user interface, and many other new packages. (*See id.* ¶¶ 61-68, 101-113; Google Mot. at 25 (only "about one-quarter" of APIs were copied).)

This case is not about Google creating a compatible platform. It is about Google picking and choosing some Java APIs, but not others, knowing it would create an *incompatible* platform. Google fragmented Java by implementing less than the full number of the Java APIs ("subsetting"), adding its own APIs ("supersetting") and, in some cases, only supporting some of

the classes within a Java API. (*See* Mitchell Opp. Report ¶¶ 61-68, 101-113.) As a result, Google's copying undermines compatibility and interoperability. Java developers have to learn new APIs to make their programs work with Android, many existing and future Java programs cannot run on Android, and many applications built for Android will not run on the Java platform or devices built to support Java. (*Id.* ¶¶ 62-68, 113.) Google's fragmentation of Java has violated the "write once, run anywhere" creed. While there had been some minor fragmentation in the past, Google recognized it was slight. (*See, e.g.*, Swoopes Decl. Ex. 11, GOOGLE-02-00111218 ("Java has very little fragmentation, and it's adoptable.").)

From the start, Sun and Oracle have offered licenses to the Java APIs for parties who agree to completely implement the APIs in a compatible way. That is a key reason why Java is so successful. But the only way to enforce this type of licensing regime is if courts enforce the copyright. Google claims it conferred a "public benefit" because it took the copyrighted works and incorporated them into the "open source Android platform." (Google Mot. at 19-20). This self-serving argument is untenable. If a party can freely ignore the copyright and fragment the platform, as Google has, compatibility in the marketplace will be severely undermined. Allowing copying that creates an incompatible end product is *against* the public interest.

The Supreme Court has emphasized that in evaluating the "character" of the use, the court should look to "'the propriety of the defendant's conduct[]'…because [f]air use presupposes 'good faith' and 'fair dealing.'" *Harper & Row Publishers, Inc. v. Nation Enters.*, 471 U.S. 539, 562 (1985) (citing 3 *Nimmer on Copyright* § 13.05[A], at 13-72). Here Google had the chance to take a license, but decided to push forward without one and "[d]o Java anyway and defend our decision, perhaps making enemies along the way." (Swoopes Decl. Ex. 8, GOOGLE-01-00019527 at 527-528; Ex. 5 at 37:1-38:23.) Google knew this was likely to hurt the licensing market for Java. (*See, e.g.*, Ex. 13, GOOGLE 26-00007930 (by creating open source platform, Sun would essentially "walk away from a $100M annual J2ME licensing business.").)

The purpose and character of Google's use weighs strongly against it. At a minimum, it raises factual issues that should go to a jury.

Nature of the Copyrighted Work – Google cites *Sega* for the proposition that "computer programs are 'essentially utilitarian'" in nature, and that, "if a work is largely functional, it receives only weak protection." (Google Mot. at 20 (citing *Sega,* 977 F.2d at 1527).) Google leaves out the line immediately above, which states: "To the extent that there are many possible ways of accomplishing a given task or fulfilling a particular market demand, the programmer's choice of program structure and design may be highly creative and idiosyncratic." *Sega*, 977 F.2d at 1524. That is the case with the Java APIs. Moreover, *Sega* found the work deserved a lower degree of protection was because it contained "unprotected aspects that cannot be examined without copying." *Sega*, 977 F.2d at 1526. There is no such issue here.

Amount and Substantiality Of The Work Used – Google argues that, quantitatively, it only used "approximately one-quarter of the API packages in the Asserted Works." (Google Mot. at 20-21.) This is a sizeable percentage and is significant quantitatively. In addition, the Supreme Court has emphasized the need to look at the "qualitative" aspect of what was used. *Harper,* 471 U.S. at 545 (copying of 300 words of quotation from unauthorized manuscript of Gerald Ford memoir was not fair use). Both parties' experts recognize the significance of the APIs. Dr. Mitchell states they are "an extremely important part of the Java platform, and are key to its organizational structure." (Mitchell Opp. Report ¶ 96; Mitchell Opening Report ¶¶ 170-173.) Dr. Astrachan claims the APIs were so important that, once Google decided to use Java, it was "essentially required" to use the APIs. (Astrachan Ex. 1 ¶¶ 130, 134-36.)

Effect Upon The Potential Market For, Or Value Of The Copyrighted Work – The Supreme Court has said that, "This last factor is undoubtedly the single most important element of fair use." *Harper*, 471 U.S. at 566-67 (citation omitted). It is also highly factual. Google asks the Court to decide this issue in its favor, as a matter of law, based on two public relations quotes from former Sun executives in 2007, when Android was first released. (Google Mot. at 21.)

"Fair use, when properly applied, is limited to copying by others which does not materially impair the marketability of the work which is copied." *Harper*, 471 U.S. at 566-67 (citation omitted). Google cannot meet this standard. It is hardly surprising that by creating a free competing platform in the mobile device arena, that gives developers the advantages of many

of Java's core APIs, Google impaired the value of the Java platform that implements them. Google substantially undermined Oracle's efforts to generate revenues. (*See* Swoopes Decl. Ex. 6 at 71:13-72:4, 111:10-112:2 (Java is "pretty well locked out of the smartphone market because of Android."). Google expected this would occur. (*See, e.g.*, Ex. 13, GOOGLE 26-00007930.)

In addition, as discussed above, Google has significantly harmed the value of the APIs by fragmenting Java and undercutting its "write once, run anywhere" capability. (*See, e.g.*, Mitchell Opp. Report ¶¶ 101-115.) Google knows this is harmful. Google itself imposes licensing terms requiring Android users to refrain from fragmenting its APIs. (*See* Swoopes Decl. Exs. 23-25.)

Google also contends that because Apache Harmony and GNU Classpath offer open source versions of the Java APIs, the inclusion of the Java APIs in Android could not be harmful. (Google Mot. at 22.) But Google knows that Harmony did not obtain a needed license from Sun (*see, e.g.*, Swoopes Decl. Ex. 1 at 119:12-122:4), and that Classpath is licensed to others under the GPL, which has important restrictions that limit its commercial application.

II. GOOGLE AND ITS AGENTS LITERALLY COPIED MULTIPLE PROGRAMS, SOURCE CODE SECTIONS, AND COMMENTS.

Google does not deny that it literally copied source code, object code and comments from 12 separate files. Google's only defense is to claim its copying is de minimis because it is insignificant in relation to Oracle's "*work as a whole*." (Google Mot. at 22-23 (quoting *Newton v. Diamond*, 388 F.3d 1189, 1195 (9th Cir. 2004) (emphasis supplied by Google).)

Google treats all of J2SE 5.0 as a single work, and compares the files to J2SE 5.0 in its entirety. (*Id.* at 23.) But each of the source code files is a separate work under copyright law. Copyright regulations allow for a single application for a "single work," which for published works is defined as "all copyrightable elements that are otherwise recognizable as self-contained works, that are included in a single unit of publication, and in which the copyright claimant is the same." 37 C.F.R. § 202.3(b)(4)(A); *see, e.g.*, *Bean v. Littell*, 669 F. Supp. 2d 1031, 1034 (D. Ariz. 2008) ("When a claimant registers a collective work, the copyright protection can also

extend to each constituent part of that work."). If this were not so, programmers could steal files at will from large software programs, hiding under the de minimis cover.

Here, each of the twelve source code files is an individual program file and is recognizable as a self-contained work that was included within J2SE 5.0, the single unit of publication. Put in its proper frame of reference, the copying is undeniably significant.

First, Google copied eight machine code files ***in their entirety***. (Mitchell Opening Report ¶¶ 241-248, Exs. Copyright-J-Q.) There is no question of significance to the work as a whole. Google copied the entire work. Moreover, Google's quotations from *Newton* are taken from a portion of the opinion analyzing infringement under the "fragmented literal similarity" test, which is only applied when "the defendant copies a portion of the plaintiff's work exactly or nearly exactly, without appropriating the work's overall essence or structure." *Newton*, 388 F.3d at 1195. (*See also* Google Mot. at 22-23.) That is not the case here. The whole work was taken.

Second, two Android source code files include a copied method that is identical right down to the spacing of the "rangeCheck" method in Oracle's Arrays.java. (Mitchell Opening Report ¶¶ 233-240.) rangeCheck is only nine lines long, but it is qualitatively significant to Arrays.java, the Oracle file in which it is located, and which is a separate work. While Arrays.java comprises 3,180 lines of code, rangeCheck "is called nine times" by other methods in the class, showing this small piece of code is important. (*Id.* ¶ 235.) In contrast, in *Newton*, the accused infringer had sampled a qualitatively insignificant portion of a four-and-a-half-minute composition ─ just three notes. *Newton*, 388 F.3d at 1191. Dr. Astrachan dismisses rangeCheck as a mundane "sanity check," but it appears to make some atypical choices in its reporting of error conditions. (*See* Mitchell Opp. Report ¶ 86.) A reasonable juror could find it is not as "common, trite, and generic" as a three-note sequence (C-D flat-C) was in *Newton*.

Third, two other Android files contain comments that are nearly identical to comments in corresponding Java files that are again separate works. (Mitchell Opening Report ¶¶ 249-251.) These two Android test programs were written to test the functionality of Android's implementations of two classes from the java.security API. The comments explain how the code tests the functionality of Android's implementation of java.security API. Comments, because

they are not "functional," can represent one of the most expressive elements of a computer program. (*Id.* ¶ 249.) They are significant to the works from which they were taken.

But even if the Court were to look beyond the individual works to measure the significance of Google's copying, summary judgment would still be inappropriate. A "use is de minimis only if the average audience would not recognize the appropriation." *Newton*, 388 F.3d at 1193. Given Google's nearly identical line-by-line copying of hundreds of lines of code, there is at least a triable issue of fact as to whether the average audience would recognize the appropriation. *Id.* at 1189. And in evaluating whether the copying was de minimis, all the copied works would have to be considered collectively, including the 37 APIs, which are "an extremely important part of the Java platform, and are key to its organizational structure." (Mitchell Opp. Report ¶ 96.) Google does not get to separate out one part of its infringement from another for the purpose of evaluating the significance of its copying.

The importance of the test code is also a disputed material fact. Dr. Mitchell rejects Dr. Astrachan's conclusion that some of these files are merely "dummy files" for testing. (Mitchell Opp. Report ¶¶ 91-95.) These files were not contained in the "test" area on Oracle's directory. But even if Google used them as test files, test files are still valuable: "testing software often requires more time and effort in software companies than writing the code initially." (*Id.* ¶ 95.) Google paid a software engineering firm $900,000 to develop a software test suite for the Java core libraries. (*See* Swoopes Decl. Ex. 15, Google-Noser Statement of Work: CTS, GOOGLE-00392183-94.) A jury could reasonably conclude that the copying of entire copyrighted files into Google's test suite was qualitatively significant.

CONCLUSION

For all the above reasons, this motion should be denied.

Dated: August 19, 2011

MORRISON & FOERSTER LLP

By: /s/ *Kenneth A. Kuwayti*
Kenneth A. Kuwayti
Attorneys for Plaintiff
ORACLE AMERICA, INC.